**REMARKS**

Applicant is in receipt of the Office Action mailed March 9, 2005. Claims 1-39 were rejected. Claims 15, 18, 24, and 25 have been amended. New claims 40-47 have been added. Claims 1-47 are currently pending in the application.

Claims 1-39 were rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,301,301 to Kodosky et al. (hereinafter "Kodosky"). Applicant respectfully traverses this rejection.

Kodosky does not teach numerous elements of the present claims. For example, claim 1 reads as follows:

> 1. (Original) A computer-implemented method for associating a first block diagram with a user interface element, the method comprising:
>
> displaying the user interface element;
>
> receiving user input specifying the first block diagram to associate with the user interface element, wherein the first block diagram includes a plurality of nodes visually indicating functionality of the user interface element; and
>
> associating the first block diagram with the user interface element, wherein the first block diagram is operable to control functionality of the user interface element.

As Applicant argued in the response to the previous Office Action, Kodosky does not teach associating a block diagram with a user interface element, wherein the block diagram includes a plurality of nodes visually indicating functionality of the user interface element and wherein the block diagram associated with the user interface element is operable to control functionality of the user interface element.

The Office Action states that:

> "Receiving user input specifying the first block diagram to associated with the user interface element, wherein the first block diagram includes a plurality of nodes visually indicating functionality of the user interface element is taught by Kodosky as the technique of a virtual instrument building block by defining an icon and connector for a virtual instrument. In defining the connector for user associate each terminal of the connector with an indicator or control on the front panel of the VI (see col. 4, lines 16-22) and once the icon and the connector have been constructed, it is then possible to use the VI as a node in a diagram (see col. 4, lines 23-25 and see Figs. 2 for nodes icon 44, 48, and 50). For example, Fig. 3 is a screen display showing an exemplary data flow block diagram which can be produced by the VI system (see col. 2, lines 23-25);"

The Examiner is correct that Kodosky teaches that a LabVIEW user can create a virtual instrument (VI) building block which can be used as a node in a block diagram (Col. 4, lines 17-38). As well known to those familiar with the LabVIEW graphical programming system, a VI has a connector which can include input and/or output terminals. A VI also has a front panel which can include controls for providing input to the VI and/or indicators for receiving output from the VI. In defining the connector for the VI, the user associates each terminal of the connector with an indicator or control on the front panel (see Col. 4, lines 17-23). For example, if the user associates a terminal with a control on the front panel then that terminal acts as an input terminal by which data is passed from the control to the VI. Similarly, if the user associates a terminal with an indicator on the front panel then that terminal acts as an output terminal by which data is passed from the VI to the indicator.

Once the icon and connector have been constructed it is then possible to use the VI as a node in a block diagram of a higher level VI. When the VI is included in the higher level VI, the icon for the VI appears in the block diagram. The user can then connect wires to input/output terminals on the icon, i.e., the input/output terminals defined when the user defined the connector for the VI. For example, Kodosky states that, "Typically the user will need to refresh his memory about the location of the inputs and outputs on the icon. LabVIEW has a 'help' feature which displays in another window the icon connection diagram: the icon with short wires attached to each input and output and labelled with the name of the associated front panel control or indicator" (see Col. 4, lines 23-32).

In addition to defining the icon and connector for the VI, the user also creates a block diagram for the VI. The block diagram includes a plurality of nodes that visually indicate functionality of the VI. For example, Fig. 3 illustrates an exemplary block diagram for a VI. If the VI is included as a node in a block diagram of a higher level VI then the block diagram of the VI may execute when the node is executed during execution of the block diagram of the higher level VI.

The block diagram of the VI visually indicates that data is received from the front panel controls and data is passed to the front panel indicators, as defined by the connector for the VI. Each front panel control has a corresponding terminal in the block diagram by

which data from the front panel control is received. For example, the block diagram of Fig. 3 illustrates a terminal labeled "Temperature" which corresponds to a control on the front panel of the VI. The front panel of the VI may be displayed as a user interface panel, and the user can interact with the "Temperature" control in order to specify data to be provided to the VI.

Similarly, each front panel indicator has a corresponding terminal in the block diagram by which data is received from the block diagram and passed to the front panel indicator. For example, the block diagram of Fig. 3 illustrates a terminal labeled "Statistic Indicators" which corresponds to an indicator on the front panel of the VI. The front panel of the VI may be displayed as a user interface panel, and the user can view the data received by the "Statistic Indicators" indicators.

Thus, in Kodsoky's graphical programming system, the block diagram of a VI may visually indicate that data is received from a front panel control user interface element and may visually indicate that data is passed to a front panel indicator user interface element. However, this is not at all the same as a block diagram visually indicating functionality of a user interface element or controlling functionality of a user interface element. In the Kodosky '301 patent, various kinds of primitive front panel controls and indicators are built in to the system and can be used in a VI, but the '301 patent does not teach or suggest any way of changing or controlling the functionality of these controls and indicators. For example, in the block diagram for a VI, the user may connect a data wire to a terminal corresponding to a front panel indicator, and the front panel indicator may display the data, but the manner in which the data is displayed cannot be changed or controlled. The '301 patent simply does not teach or suggest any capability that allows users to affect the manner in which an indicator displays data or to affect other aspects of the functionality of a control or indicator.

In contrast, claim 1 recites that a block diagram may be associated with a user interface element in response to user input such that the block diagram visually indicates functionality of the user interface element and such that the block diagram is operable to control functionality of the user interface element. Thus, a user may create a block diagram that visually indicates functionality of a user interface element and may associate the block diagram with the user interface element so that the block diagram is operable to

control functionality of the user interface element. These features are simply not taught by the Kodosky '301 patent.

With regard to the element in claim 1 of, "wherein the first block diagram is operable to control functionality of the user interface element," the Examiner refers to Col. 8, lines 53-55, which teaches, "a library of function icons, each representing a mathematical operation to be performed on specified input data to generate output data". The Examiner asserts that, "Thus, a mathematical input operation of each of plurality icons will, in turn, control the functionality and performance of output data." Thus, the Examiner refers to controlling the functionality and performance of output data, but Applicant notes that claim 1 does not recite controlling the functionality of output data, but instead recites, "wherein the first block diagram is operable to control functionality of the user interface element". A user interface element is not at all the same as output data. Furthermore, Applicant respectfully submits that it does not make sense to assert that a mathematical operation controls the functionality and performance of output data. As taught in Kodosky, a mathematical operation may be performed on specified input data to generate output data. However, output data is simply data, and data does not have "functionality" or "performance".

Applicant thus submits that claim 1 is allowable over Kodosky, for at least the reasons discussed above.

Taking amended claim 15 as another exemplary independent claim in the present application, the claim recites:

> 15. (Currently Amended) A computer-implemented method for including a user interface element in a graphical program, the method comprising:
>
> receiving user input specifying inclusion of the user interface element in the graphical program, wherein the user interface element has an associated first block diagram that implements first functionality of the user interface element, wherein the graphical program includes a second block diagram that implements second functionality of the graphical program, wherein the second block diagram is separate from the first block diagram associated with the user interface element;
>
> including the user interface element in the graphical program in response to the user input;
>
> wherein said including the user interface element in the graphical program comprises including the first block diagram associated with the user interface element in the graphical program;

wherein, during execution of the graphical program, the first block diagram associated with the user interface element is operable to control the first functionality of the user interface element and the second block diagram is operable to control the second functionality of the graphical program.

Thus, as recited in claim 15, the graphical program includes a second block diagram (which may be referred to as a main block diagram) that implements second functionality of the graphical program. The block diagram associated with the user interface element implements first functionality of the user interface element and is a separate block diagram from the main block diagram of the graphical program. Kodosky simply does not teach or suggest a user interface element that has an associated block diagram, where the user interface element can be included in a graphical program that has a main block diagram, and where the block diagram associated with the user interface element is separate from the main block diagram of the graphical program.

Applicant notes that claim 12, which is dependent upon claim 1, and claim 13, which is dependent upon claim 12 recite similar elements of, "including the user interface element in a graphical program in response to user input," and "wherein the graphical program includes a second block diagram; wherein the second block diagram is separate from the first block diagram" (i.e., where the first block diagram is associated with the user interface element). In rejecting claim 13, the Office Action states that, "the limitations of wherein the graphical program includes a second block diagram and wherein the second block diagram is separate from the first block diagram are taught by Kodosky as the technique of in LabVIEW, the conversion is does automatically, where necessary, in the dataflow block diagrams (see col. 3 line 68 to col. 4 line 1) and LabVIEW user creates VIs which can be used as building blocks in other VIs (see col. 4, lines 54-55)."

However, col. 3 line 68 to col. 4 line 1 pertains to conversions performed on inputs to polymorphic functions. In other words, polymorphic data conversion is performed automatically where necessary in dataflow block diagrams that utilize polymorphic functions. However, Applicant submits that this has little or no relevance to the features recited in claim 13 wherein a first block diagram is associated with a user interface element and is operable to control functionality of the user interface element,

and wherein the user interface element is included in a graphical program that has a second block diagram that is separate from the first block diagram associated with the user interface element.

As per the reference to LabVIEW users creating VIs that can be used as building blocks in other VIs (col. 4, lines 54-55), this aspect of Kodosky has already been discussed above. A user can indeed create a VI that can be included in the block diagram of a higher level VI, but Kodosky simply does not teach or suggest the concept of including a user interface element (which is not the same as a VI) in a graphical program that includes a second block diagram, wherein the user interface element has an associated first block diagram that is separate from the second block diagram.

Applicant thus submits that claims 13 and 15 are allowable over Kodosky, for at least the reasons discussed above. Inasmuch as the other independent claims recite similar elements as those discussed above, Applicant submits that the other independent claims are also allowable over Kodosky.

Since the independent claims have been shown to be allowable over Kodosky, Applicant submits that the dependent claims are also allowable over Kodosky for at least this reason. Applicant also respectfully submits that numerous ones of the dependent claims recite further distinctions over Kodosky.

For example, claim 27 recites the additional limitation of, "wherein the first block diagram [i.e., the block diagram associated with the user interface element] is operable to change a manner in which data is displayed in the user interface element." In the rejection of claim 27 the Office Action states that this limitation "is taught by Kodosky as the technique of LabVIEW user crates VIS which can be used as building blocks in other VIs. VIs are analogous to subroutines so it is useful to display the hierarchical relationship of VIs. LabVIEW automatically constructs a diagram showing the hierarchical of all VIs in memory (see col. 4, lines 54-59)."

However, displaying a diagram that shows a hierarchical relationship of VIs is not at all the same as what is actually recited in claim 27, i.e., "wherein the first block diagram is operable to change a manner in which data is displayed in the user interface element." The Office Action has apparently taken the user interface element recited in claim 1 to mean a control or indicator in the front panel of a VI as taught in Kodosky.

However, the hierarchical diagram referred to in the above-cited portion of Kodosky is displayed in a separate window or panel, not in a front panel control or indicator. See Fig. 7 for an example of a diagram that shows a hierarchical relationship of VIs. Moreover, the hierarchical diagram is not displayed by a block diagram, and in particular, is not displayed by a block diagram associated with a user interface element. Applicant thus submits that claim 27 is allowable.

As another example, claim 28 reads as follows:

> 28. (Previously Presented) The method of claim 1, further comprising:
> copying the user interface element from a first graphical program to a second graphical program;
> programmatically including the first block diagram in the second graphical program in response to said copying.

The element of "programmatically including the first block diagram in the second graphical program in response to said copying" is simply not taught by Kodosky. In the rejection of claim 28, the Office Action refers to "selecting the Graphical Array and Graph from Functional Palette (see col. 8, lines 31-33 and see Fig. 19)". However, Fig. 19 merely shows an Array and Graph Function palette that includes various function nodes which the user can include in the block diagram of a VI. Including a function node from a palette into a VI is not at all the same as copying a user interface element from a first graphical program to a second graphical program, as recited in claim 28. Applicant thus submits that claim 28 is allowable.

Applicant also notes that various new dependent claims that recite additional limitations have been added. For example, new claim 41 reads as follows:

> 41. (New) The method of claim 15,
> wherein the main block diagram is operable to provide data to the user interface element;
> wherein the block diagram associated with the user interface element is operable to receive the data from the main block diagram and control a visual appearance of the user interface element based on the data.

Applicant respectfully submits that Kodosky does not teach a block diagram associated with a user interface element receiving data from a main block diagram of a graphical program and controlling a visual appearance of the user interface element based on the data. Applicant thus submits that claim 41 is allowable.

## CONCLUSION

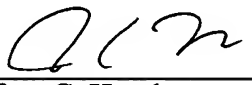
In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-46000/JCH.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

Respectfully submitted,



Jeffrey C. Hood
Reg. No. 35,198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 6/7/2005 JCH/JLB